Parameter-free HE-friendly Logistic Regression

Anonymous Author(s)
Affiliation
Address
email

Abstract

Privacy in machine learning has been widely recognized as an essentially ethical and legal standard, because the data used for machine learning may contain individuals' sensitive information. Homomorphic encryption has recently attracted attention as a key solution for privacy-preserving machine learning, but previous approaches on the training of encrypted machine learning have relied heavily on hyperparameter selection, which should be avoided because it is impossible to conduct validation on encrypted data. In this paper, we propose an effective privacy-preserving logistic regression method that is free from the approximation of sigmoid function as well as hyperparameter selection. In our framework, a logistic regression model can be turned into the corresponding Ridge regression for the logit. We provide a theoretical background of our framework by suggesting a new generalization error bound on the encrypted data. The experiments on various real-world data show that our framework achieves better classification results while saving $\sim 68\%$ of latency, compared to the previous models.

1 Introduction

Machine Learning on Encrypted Data (MLED) is an effective method to ensure privacy for both machine learning models and the data used for the training of the models. Unlike other privacy preserving machine learning methods, MLED does not require decryption of intermediate products of the algorithm, and data transfer between participants does not occur, enabling complete outsourcing of machine learning. As the value of data increases, MLED models have been actively studied as an attempt to combine data and computing power that are separated from each other. With the development of efficient Homomorphic Encryption (HE) schemes that enable MLED, recent studies [3, 7, 30, 21, 22, 23] cover most of the machine learning models, from simple linear regression to deep neural networks (DNN).

However, current research on MLED is still struggling because of the limitations of the proposed HE schemes. Operations on ciphertexts are very inefficient in terms of computation time and memory consumption, compared to operations on plaintexts. In addition, after performing a certain amount of operations, a costly procedure called bootstrapping is required [14, 5]. Therefore, the majority of the studies are limited to the inference phase, assuming a situation that provides machine learning inference as a service. To produce the maximum value from the data, the training phase should be studied more deeply.

One of the MLED models whose training steps were studied is logistic regression (LR). LR is a simple classification model that linearly divides the data space, which is still used effectively in a variety of fields including the medical field, marketing, and geology [6, 8, 20]. Kim et al. [18] trained encrypted LR with gradient descent and approximated the sigmoid function with the most similar polynomials within a certain range. While performing well on several benchmark datasets, their methodology raises some concerns, because the training parameters such as learning rate, the number

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

of iterations, and the range of approximating the sigmoid function should be determined in advance. Indeed, the problem is common for all proposed MLED training models.

In this study, we propose an effective privacy-preserving LR method that is free from hyperparameter selection. We first train an unencrypted classification model to extract the prediction probability for each label. Then we solve for a Ridge regression that predicts the logit of the estimated probability. Our theoretical results provide a new generalization error bound, which can be optimized by both properly estimating the logit and minimizing the loss between the logit and the output of the regression. To benefit from the trade-off between security and efficiency, we admit the presence of private variables, whose privacy is more important than the privacy for others, which has been widely recognized in other fields but has been rarely introduced into MLED. The experimental results show that compared to the prior encrypted LR model, our method shows both better classification results and lower training latency.

2 Preliminary

Throughout this paper, we write vectors in bold lowercase and matrices in bold uppercase. A ciphertext is indicated in the form of the corresponding message enclosed in brackets (e.g. $[m]$). Operations between ciphertexts or between ciphertext and plaintext are both represented as those between plaintexts.

Fully Homomorphic Encryption A fully HE (FHE) scheme consists of 4 procedures : $\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}$ and Eval. KGen generates secret key sk and public key pk [13]. A message m is encrypted to a ciphertext $[m] = \mathsf{Enc}(m, pk)$, and a ciphertext $[m']$ is decrypted back to a plaintext $m' = \mathsf{Dec}([m'], sk)$. Eval evaluates a function f with a vector of ciphertexts $([m_1], \ldots, [m_k])$ as an input, so it holds that $\mathsf{Dec}(\mathsf{Eval}(([m_1], \ldots, [m_k]), f, pk), sk) = f(m_1, \ldots, m_k)$. An FHE scheme naturally enables addition and multiplication, so non-polynomial operations must be approximated by polynomial operations. A noise is introduced when a message is first encrypted, and it grows as the operation between ciphertexts proceeds. If the noise exceeds a certain level, the ciphertext cannot be correctly decrypted. FHE allows an unlimited number of operations through bootstrapping which resets the noise, but bootstrapping requires a large computational cost. Therefore, leveled HE (LHE) which avoids bootstrapping is preferred in many studies, with the number of operations required and the corresponding scheme parameters determined in advance. In the case of using gradient descent in MLED training, LHE is not suitable because the number of iterations is very limited, whereas our method is suitable for LHE because the number of required operations is not too large and can be determined before implementation.

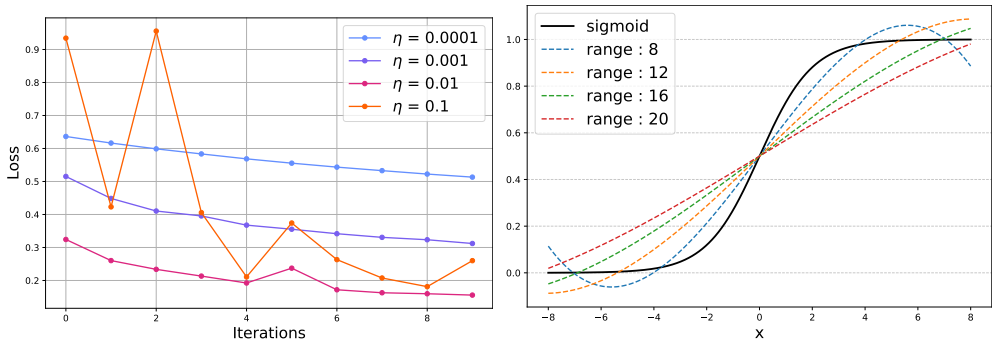
Logistic Regression with HE The loss function of LR is the negative log-likelihood, which is given by

$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \log(1 + \exp(-y^i \boldsymbol{\theta}^T \boldsymbol{x}^i)) \tag{1}$$

where $\boldsymbol{\theta}$ is the weight vector, $\boldsymbol{x}^i$ is i-th input and $y^i \in \{+1, -1\}$ is the label of i-th input. Kim et al. [17] minimized (1) with Nesterov's accelerated gradient, which is a slight modification of the gradient descent. The gradient descent can be written as follows:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t, \quad \boldsymbol{g}_t = \frac{\partial J}{\partial \boldsymbol{\theta}} = -\sum_{i=1}^{n} \mathrm{sig}(-y^i \boldsymbol{\theta}_t^T \boldsymbol{x}^i) \cdot y^i \boldsymbol{x}^i \tag{2}$$

where η is the learning rate and $\mathrm{sig}(x) = 1/(1+\exp(-x))$ is the sigmoid function. When implementing their model with HE, there are two major limitations. The first is the selection of the learning rate and the number of iterations of the gradient descent. In Figure 1(a) we plotted how the loss changes according to the number of iterations for different learning rates. It is shown that the performance can be very different depending on the learning rate. Also, the optimal learning rate is different depending on the number of iterations (compare orange line ($\eta = 0.1$) and pink line ($\eta = 0.01$)). When dealing with plaintext, the optimal parameters can be found through cross validation, but this is impossible in MLED because the modeler cannot decrypt the weight vector. On the other hand, our method can obtain the closed-form solution without using gradient descent.



(a) Importance of choosing the appropriate parameters in learning LR

(b) Approximation performance of sigmoid function decreases as possible range of input grows

Figure 1: Sensitivity analysis on the hyperparameters in previous privacy preserving LR

Another limitation is that the sigmoid function cannot be efficiently evaluated with HE. Because the low-order Taylor expansion only gives a good approximation in a very localized domain, Kim et al. [17] approximated the sigmoid function with a polynomial showing the lowest mean squared error in the domain $[-8, 8]$. Figure 1(b) shows the different approximation results according to the domain range when approximated by 3rd order polynomial. The maximum error between the polynomial and the sigmoid function is 0.11 when the range is $[-8, 8]$ (blue line), but increases to 0.26 when the range is $[-20, 20]$ (red line). Therefore, it is essential to obtain a tight bound for the input value of the sigmoid function, but it is not possible for encrypted data. Conversely, our method solves Ridge regression on the ciphertext, so there is no need to evaluate the sigmoid function.

Partial Encryption of Private Variables It has been widely accepted that privacy for some variables is more important than security for other variables. In the field of privacy-preserving data mining, k-anonymity [32] transforms the variables to be published to prevent exposure of personal identity. Subsequent concepts such as l-diversity [24, 33] and t-closeness [19] were proposed to protect other private variables, including personal income and medical record. In fair machine learning studies which have attracted a lot of interest in recent years, variables that may cause discrimination are defined as sensitive variables, and it is required that sensitive variables not be disclosed [15, 16]. In cryptography research, Sumathi and Sangeetha [31], Gai et al. [10] presented security criteria and variables were selectively encrypted accordingly.

When partially encrypting some private variables, the main purpose is to achieve a trade-off between security and computation & storage efficiency. In Chapter 3 we propose an efficient Ridge regression method with a partially encrypted dataset, which cannot be directly applied to LR. We separate operations on encrypted and non-encrypted variables, whereas in LR, the input of the sigmoid function is the weighted sum of all variables. In addition, simply applying the existing LR algorithm to partially encrypted data does not bring sufficient efficiency, nor does it solve the problems of parameter selection. We solve this problem by replacing the LR with the Ridge regression for the pre-calculated logit.

3 Framework

In this paper, we propose a new framework that can effectively handle classification tasks, while preserving the privacy of private variables. We set two different datasets of the same variables, $\mathcal{D}_1 = \{(\boldsymbol{x}_1^i, y_1^i) | i = 1, \ldots, n_1\}$ and $\mathcal{D}_2 = \{(\boldsymbol{x}_2^j, y_2^j) | j = 1, \ldots, n_2\}$, where $\boldsymbol{x}_2^j = (x_{21}^j, \ldots, x_{2p}^j, [x_{2(p+1)}^j], \ldots, [x_{2(p+\ell)}^j])$ contains ℓ encrypted private variables.

Threat Model The participants of our framework consists of data owners O, a modeler M and a crypto-service provider C. We assume that M and C are honest-but-curious and do not collude, which is widely accepted in MLED studies [29, 26, 1, 27]. Our security goals are as follows:

- Neither M nor C should obtain information on private variables.

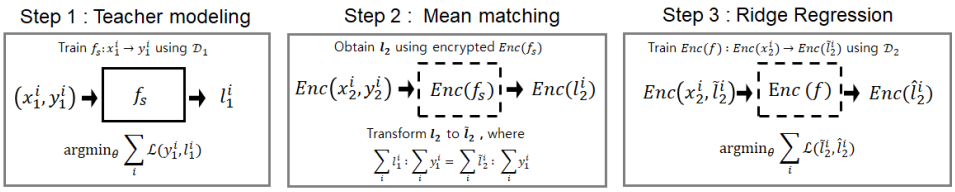


Figure 2: Overall framework of our proposed method

- Neither O nor C should obtain information on the learned model.

The details of our protocol to achieve the security goal is in Appendix.

Main Framework Our framework consists of three steps. In the first step, M trains a classification model with $\mathcal{D}_1$, which mimics the first phase of the knowledge distillation in that with the first model we extract the soft probability from the target label. However, our method has several major differences from knowledge distillation. First, knowledge distillation aims to train a rather simple model that performs comparable to a complex teacher model, and extracting the probability is a means to achieve the goal. On the other hand, in our framework the probability plays a more important role in that we change the classification task into a regression problem with it. Another difference is that in our method, the teacher model need not be a complex model. There are several advantages to using a simple model as a teacher model, which we will demonstrate later.

Meanwhile, we should assume that M should possess unencrypted $\mathcal{D}_1$ that can be used for step 1. Considering that M is depicted as a company seeking to profit from their model in many studies [21, 22, 23], there may be individuals who provide their private information to M in return. To make a more realistic assumption, we set there are relatively few people who publish their private information, and the underlying distributions of $\mathcal{D}_1$ and $\mathcal{D}_2$ are heterogeneous because they have different features than those who do not want to provide private information. To fit to the former setting, we use a simple model that generalize well enough with a small amount of data as the teacher model.

In the second step, using the teacher model M infers the logit of $\mathcal{D}_2$. In the case where the teacher model is LR, the inference is simply an inner product between the model parameter $\boldsymbol{\theta}$ and $\boldsymbol{x}^j$s, which can be efficiently computed with HE using slot-wise rotation (See the Allsum algorithm in [18]). Denoting the teacher model as f_s and the logit of $\mathcal{D}_1$ as $\boldsymbol{l}_1$, and $\mathcal{D}_2$ as $\boldsymbol{l}_2$, we can obtain the $\boldsymbol{l}_2$ from $\mathcal{D}_2$ using the teacher model f_s. However, because $\mathcal{D}_1$ and $\mathcal{D}_2$ have different distributions, so may $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. Therefore, rather than directly using the logit distribution $\boldsymbol{l}_2$, we suggest adding a regularization term β on $\boldsymbol{l}_2$ so that it can take into account the difference between two distributions. The β should satisfy:

$$\frac{\sum_i f_s(x_1^i)}{\sum_i y_1^i} = \frac{\sum_i f_s(x_2^i) + \beta}{\sum_i y_2^i} \tag{3}$$

Now applying the regularization term β, we can obtain the shifted logit $\tilde{\boldsymbol{l}}_2$, where $\tilde{l}_2^i = f_s(x_2^i) + \beta$.

Finally, we train a ridge regression on $\mathcal{D}_2$, where the target variable is the shifted logit $\tilde{\boldsymbol{l}}_2$, not $\boldsymbol{y}_2$. Intuitively, using a well-estimated probability enables at least as accurate training as using a binary target, and our theoretical results in Chapter 4 supports this claim. By encrypting a small private portion of the whole information, we can train an efficient ridge regression without having to explore the learning parameters. Let $\boldsymbol{X} = ((\boldsymbol{x}_2^1)^T \cdots (\boldsymbol{x}_2^{n_2})^T)^T \in \mathbb{R}^{n_2 \times (p+\ell)}$, $[\boldsymbol{H}_s] = ([\boldsymbol{h}_1] \cdots [\boldsymbol{h}_\ell])$ the encrypted ℓ columns of $\boldsymbol{X}$ and $\boldsymbol{X}_{(-s)}$ the other p columns. Then the ridge estimate is

$$\hat{\boldsymbol{f}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda \boldsymbol{I}_{p+\ell})^{-1}\boldsymbol{X}^T\tilde{\boldsymbol{l}}_2 = \boldsymbol{X}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_{n_2})^{-1}\tilde{\boldsymbol{l}}_2 = \tilde{\boldsymbol{l}}_2 - \lambda(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_{n_2})^{-1}\tilde{\boldsymbol{l}}_2. \tag{4}$$

Because $\boldsymbol{X}\boldsymbol{X}^T = \boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + [\boldsymbol{H}_s][\boldsymbol{H}_s^T]$, by applying the Sherman-Woodbury inversion formula we have

$$(\boldsymbol{X}\boldsymbol{X}^T + \lambda \boldsymbol{I}_{n_2})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}[\boldsymbol{H}_s](\boldsymbol{I}_\ell + [\boldsymbol{H}_s^T]\boldsymbol{A}^{-1}[\boldsymbol{H}_s])^{-1}[\boldsymbol{H}_s^T]\boldsymbol{A}^{-1}$$

where $\boldsymbol{A} = \boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_{n_2}$. Using the singular value decomposition (SVD) of $\boldsymbol{X}_{(-s)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ where $\boldsymbol{U} \in \mathbb{R}^{n_2 \times n_2}$ and $\boldsymbol{V} \in \mathbb{R}^{p \times p}$ are orothogonal matrices and $\boldsymbol{\Sigma} \in \mathbb{R}^{n_2 \times p}$ are diagonal matrix with diagonal entries $\sigma_1 \geq .... \geq \sigma_p \geq \sigma_{p+1} = \cdots = \sigma_{n_2} = 0$, we have

$$\boldsymbol{A}^{-1} = (\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\boldsymbol{U}^T + \lambda \boldsymbol{I}_{n_2})^{-1} = \boldsymbol{U}(\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T + \lambda \boldsymbol{I}_{n_2})^{-1}\boldsymbol{U}^T.$$

Then the ridge estimate can be rewritten as

$$\hat{\boldsymbol{f}} = \tilde{\boldsymbol{l}}_2 - \lambda \boldsymbol{A}^{-1}\tilde{\boldsymbol{l}}_2 + \lambda \left(\sum_{j=1}^{n_2} \frac{\boldsymbol{u}_j \boldsymbol{u}_j^T [\boldsymbol{H}_s]}{\sigma_j^2 + \lambda} \right) (\boldsymbol{I}_\ell + [\boldsymbol{\Xi}])^{-1} \left(\sum_{j=1}^{n_2} \frac{1}{\sigma_j^2 + \lambda} [\boldsymbol{H}_s^T] \boldsymbol{u}_j \boldsymbol{u}_j^T \tilde{\boldsymbol{l}}_2 \right) \quad (5)$$

where the $\boldsymbol{u}_j$ are the columns of $\boldsymbol{U}$ and $[\boldsymbol{\Xi}] = \sum_{j=1}^{n_2} \frac{1}{\sigma_j^2+\lambda}[\boldsymbol{H}_s^T]\boldsymbol{u}_j\boldsymbol{u}_j^T[\boldsymbol{H}_s]$. The result implies that, by separating the operations that involve encrypted data from those that do not, we can reduce the dimension of the matrix inversion from $(p+\ell) \times (p+\ell)$ to $\ell \times \ell$. Using Newton-Schulz iterative algorithm [2] for matrix inversion, multiplications between $(p+\ell) \times (p+\ell)$ matrices reduces to multiplications between $\ell \times \ell$ matrices, which gives quadratic reduction in the computation cost. Moreover, it is known that Newton-Schulz type methods can fail with large matrices [12], so reducing the size of the matrix greatly improves the stability. It is notable that when $\ell = 1$ or 2, the matrix inverse reduces to a scalar inverse, and can be more easily calculated with Goldschmidt's division algorithm [11].

However, there is still a problem of latency because the computation of (5) involves n_2 summations. To simplify the computations, we notice that $\boldsymbol{U} = [\boldsymbol{U}_1, \boldsymbol{U}_2] \in \mathbb{R}^{n_2 \times n_2}$ where $\boldsymbol{U}_1 = [\boldsymbol{u}_1, ..., \boldsymbol{u}_p] \in \mathbb{R}^{n_2 \times p}$ and $\boldsymbol{U}_2 = [\boldsymbol{u}_{p+1}, ..., \boldsymbol{u}_{n_2}] \in \mathbb{R}^{n_2 \times (n_2 - p)}$ and $\boldsymbol{U}_1 \perp \boldsymbol{U}_2$. Using the reduced SVD and $\sigma_{p+1} = \cdots = \sigma_{n_2} = 0$, we have the freedom to choose $\boldsymbol{U}_2$ as long as $\boldsymbol{U}_1 \perp \boldsymbol{U}_2$ in such a way that the orthogonalization process is applied to the order of $\boldsymbol{h}_k$, $k = 1, ..., \ell$.

- $\boldsymbol{u}_j$ are orthogonal to $\boldsymbol{h}_k$, for all $j = p+k+1, ..., n_2$
- $\boldsymbol{u}_{p+k}$ are the complements of the orthogonal projections of $\boldsymbol{h}_k$, onto $[\boldsymbol{U}_1, \boldsymbol{u}_{p+k-1}]$, $k = 1, ..., \ell$, respectively and orthogonal to each other.

That is, for $k = 1, ..., \ell$,

$$\hat{\boldsymbol{u}}_{p+k} = \boldsymbol{h}_k - \sum_{i=1}^{p+k-1} \boldsymbol{u}_i(\boldsymbol{u}_i^T \boldsymbol{h}_k), \quad \boldsymbol{u}_{p+k} = \hat{\boldsymbol{u}}_{p+k}/\|\hat{\boldsymbol{u}}_{p+k}\|.$$

Then $[\boldsymbol{\Xi}]$ and $\hat{\boldsymbol{f}}$ can be further simplified as

$$[\boldsymbol{\Xi}] = \sum_{j=1}^{p+\ell} \frac{1}{\sigma_j^2 + \lambda} [\boldsymbol{H}_s^T] \boldsymbol{u}_j \boldsymbol{u}_j^T [\boldsymbol{H}_s],$$

$$\hat{\boldsymbol{f}} = \tilde{\boldsymbol{l}}_2 - \lambda \boldsymbol{A}^{-1}\tilde{\boldsymbol{l}}_2 + \lambda \left(\sum_{j=1}^{p+\ell} \frac{\boldsymbol{u}_j \boldsymbol{u}_j^T [\boldsymbol{H}_s]}{\sigma_j^2 + \lambda} \right) (\boldsymbol{I}_s + [\boldsymbol{\Xi}])^{-1} \left(\sum_{j=1}^{p+\ell} \frac{1}{\sigma_j^2 + \lambda} [\boldsymbol{H}_s^T] \boldsymbol{u}_j \boldsymbol{u}_j^T \tilde{\boldsymbol{l}}_2 \right). \quad (6)$$

Algorithm Algorithm 1 describes the procedure of training Ridge regression, assuming that there is one private variable. For convenience, our method was described as obtaining the ridge estimate, but to inference about test data, the algorithm is slightly modified to output the weight vector. We assume that all n_2 samples of private variables are encoded into a single ciphertext to enable SIMD operations. If the number of the samples is larger than the maximum number of samples which can be packed into a ciphertext, we can naturally split the samples to multiple ciphertexts. Excluding addition and multiplication operations for the ciphertext, the algorithm consists of three functions, InnerProduct, Inv and MatVecProduct. InnerProduct returns a ciphertext where the values of all plaintext slots are the same as the inner product of two vectors. Inv evaluates Goldschmidt's division

Algorithm 1 Training Ridge Regression with Encrypted Private Variable

Input: unencypted data $\boldsymbol{X}_{(-s)}$, encrypted private variable $[\boldsymbol{h}_s]$, target variable $\boldsymbol{l}$
Output: two ciphertexts $[\boldsymbol{\omega}_{(-s)}]$, $[\omega_s]$ such that the weight vector of Ridge regression is $\boldsymbol{\omega}^T = (\boldsymbol{\omega}_{(-s)}^T \quad \omega_s)$
≪*Pre-computations*≫
[-] Calculate matrix $\boldsymbol{A}^{-1} = (\boldsymbol{X}_{(-s)}\boldsymbol{X}_{(-s)}^T + \lambda \boldsymbol{I}_{n_2})^{-1}$ using the collected user data $\mathbf{X}_{(-s)}$
[-] Obtain orthogonal matrices $\boldsymbol{U} = (\boldsymbol{u}_1 \cdots \boldsymbol{u}_p), \boldsymbol{V} = (\boldsymbol{v}_1 \ldots \boldsymbol{v}_p)$ and diagonal entries $\boldsymbol{\sigma} = (\sigma_1, \cdots, \sigma_p)$ using the SVD of $\boldsymbol{X}_{(-s)}$
≪*Computations on Encrypted data*≫
[-] $[c_1] \leftarrow \mathsf{InnerProduct}(\boldsymbol{A}^{-1}\boldsymbol{l}, [\boldsymbol{h}_s])$; $[\boldsymbol{c}_2] \leftarrow \mathsf{MatVecProd}(\boldsymbol{V}, \frac{\boldsymbol{\sigma}}{\boldsymbol{\sigma}^2+\lambda}, \boldsymbol{U}, [\boldsymbol{h}_s]) \cdot [c_1]$;
[-] $[\boldsymbol{c}_3] \leftarrow \mathsf{MatVecProd}(\boldsymbol{U}, \frac{1}{\boldsymbol{\sigma}^2+\lambda}, \boldsymbol{U}, [\boldsymbol{h}_s])$; $[c_4] \leftarrow \mathsf{Inv}(\mathsf{InnerProduct}([\boldsymbol{c}_3], [\boldsymbol{h}_s]) + 1)$
[-] Output $[\boldsymbol{\omega}_{(-s)}] \leftarrow \boldsymbol{X}_{(-s)}^T \boldsymbol{A}^{-1}\boldsymbol{l} - [\boldsymbol{c}_2] \cdot [c_4]$ and $[\omega_s] \leftarrow [c_1] \cdot [c_4]$

algorithm in Algorithm 1, but in the case where $s \geq 3$, it evaluates an iterative method for matrix inversion. MatVecProduct evaluates an efficient matrix-vector multiplication using the SVD of the input matrix. The detailed algorithms for these functions are in Appendix.

Extension to Nonlinear Models A limitation of our framework is that it can model only the linear relationship between the input variables and the target variable. However, when partially encrypting private variables, our method has the capacity to model the nonlinear relationship between variables using kernel methods. The formulation for the extension, and some experimental results are in Appendix.

4 Theoretical Framework

This section provides the core theory behind our proposed methodology. We provide a generalization error bound for the proposed method.

Consider a classification task where $\mathcal{X}$ is an input space and $\mathcal{Y} = \{0, 1\}$ is an output space. Let $\mathcal{Z}_S := \mathcal{X} \times \mathcal{Y}$ and $\mathcal{Z}_T = \mathcal{X} \times [0, 1]$. Notice that $\mathcal{Z}_S \subset \mathcal{Z}_T$. Let $S = \{\mathbf{z}_i = (\mathbf{x}_i, y_i) \in \mathcal{Z}_S : \ i = 1, ..., N_s\}$ represent the given open labelled data of N_s samples. In our classification settings, we have two distinct distributions: $(\mathbf{x}, y) \sim \mathcal{D}_S$ according to a sample distribution $\mathcal{D}_S$ over $\mathcal{Z}_S$ with and $(\mathbf{x}, p) \sim \mathcal{D}_T$ according to a target distribution $\mathcal{D}_T$ over $\mathcal{Z}_T$.

Utilizing these concepts and following the notations in [25], a hypothesis $h : \mathcal{Z}_S \to \Re\}$ is a scoring function such that we assign to each point $\mathbf{x}$ the class label of the maximum score $h(\mathbf{x}, y)$, that is, $\arg\max_{y \in \mathcal{Y}} h(\mathbf{x}, y)$. The *margin* $\xi_h(\mathbf{z})$ at a sample example $\mathbf{z} = (\mathbf{x}, y) \sim \mathcal{D}_S$ is then defined by

$$\xi_h(\mathbf{z}) = h(\mathbf{x}, y) - \max_{y' \neq y} h(\mathbf{x}, y')$$

Thus, h misclassifies $\mathbf{z}$ iff $\xi_h(\mathbf{z}) \leq 0$.

The generalization error (or risk) of a hypothesis $h \in \mathcal{H} := \{h : \mathcal{Z}_S \to \Re\}$ in a sample distribution $\mathcal{D}_S$ is defined by

$$\mathcal{R}_{\mathcal{D}_S}(h) = \mathbb{E}_{\mathbf{z} \sim \mathcal{D}_T}[1_{\xi_h(\mathbf{z}) \leq 0}]$$

Given a ground truth function c_T, and a hypothesis set $\tilde{\mathcal{H}} = \{\tilde{h} : \mathcal{Z}_T \to \Re\}$, the generalization error (or risk) of a hypothesis $\tilde{h} \in \tilde{\mathcal{H}}$ in a true target distribution $\mathcal{D}_T$ is defined by

$$\mathcal{R}_{\mathcal{D}_T}(\tilde{h}) = \mathcal{R}_{\mathcal{D}_T}(\tilde{h}, c_T) = \mathbb{E}_{\mathbf{z} \sim \mathcal{D}_T}[\|\tilde{h}(\mathbf{z}) - c_T(\mathbf{z})\|]$$

Notice that $\mathcal{H} \subset \tilde{\mathcal{H}}$. If we consider $\hat{\mathcal{D}}_T^N$ as $\hat{\mathcal{D}}_S^N$ samples of size N drawn independently according to the respective distributions $\mathcal{D}_S$ and $\mathcal{D}_T$, then the empirical error of $h \in \mathcal{H}$ and $\tilde{h} \in \tilde{\mathcal{H}}$ is respectively defined by

$$\hat{\mathcal{R}}_{\hat{\mathcal{D}}_S^N}(h) = \frac{1}{N} \sum_{\mathbf{z}_i \in \hat{\mathcal{D}}_S^N} 1_{\xi_h(\mathbf{z}_i) \leq 0}, \quad \text{and} \quad \hat{\mathcal{R}}_{\hat{\mathcal{D}}_T^N}(\tilde{h}) = \frac{1}{N} \sum_{\mathbf{z}_i \in \hat{\mathcal{D}}_T^N} \|\tilde{h}(\mathbf{z}_i) - c_T(\mathbf{z}_i)\|$$

We next define the $\mathcal{H}, \tilde{\mathcal{H}}$-divergence for $\mathcal{H} \subset \tilde{\mathcal{H}}$ which measures the discrepancy between two different distributions by

$$D_{\mathcal{H},\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T) = \sup_{h \in \mathcal{H}, \tilde{h} \in \tilde{\mathcal{H}}} |\mathcal{R}_{\mathcal{D}_S}(h) - \mathcal{R}_{\mathcal{D}_T}(\tilde{h})|$$

This discrepancy is measured using two different hypothesis sets $\mathcal{H}$ and $\tilde{\mathcal{H}}$, where $h \in \mathcal{H} := \{h : \mathcal{Z}_S \to \Re\}$ and $h \in \tilde{\mathcal{H}} = \{\tilde{h} : \mathcal{Z}_T \to \Re\}$. We are now ready to derive the following generalization error risk in our classification setting.

Theorem 1. *Let $\mathcal{D}_S$ and $\mathcal{D}_T$ be the sample and the true target distributions, respectively. Then for any hypothesis $\tilde{h} \in \tilde{\mathcal{H}}$, the following inequality holds:*

$$\mathcal{R}_{\mathcal{D}_T}(\tilde{h}) \leq \min_{h' \in \mathcal{H}} \mathcal{R}_{\mathcal{D}_S}(h') + D_{\mathcal{H},\tilde{\mathcal{H}}}(\mathcal{D}_S, \mathcal{D}_T) \tag{7}$$

As a result, the upper bound of empirical expected target error can be decomposed into two parts: The first term is the empirical source error. In our framework, we try to minimize this term by training a teacher classification model with $\mathcal{D}_1$ in the first step. Then the second term is the $\mathcal{H}, \tilde{\mathcal{H}}$-divergence between the $\mathcal{D}_S$ and $\mathcal{D}_T$ which can be interpreted as distillation error. The step 2 in our framework correspond to this part, it calculates the probability p in the distribution $\mathcal{D}_T$, since $(\mathbf{x}, p) \sim \mathcal{D}_T$. Finally, by training the student function in step 3, we can minimize the second term. We provide more details in Appendix.

We next present the following margin bound for classification in the probably approximately correct (PAC) learning framework.

Theorem 2. *Let $\mathcal{D}_S$ and $\mathcal{D}_T$ be the sample and the true target distributions, respectively. Then, for any $\delta > 0$, with probability at least $1 - \delta$, the following classification generalization bound holds for all hypothesis $h \in \mathcal{H}_\rho = \{(\mathbf{x}, y) \to \boldsymbol{\beta} \cdot (y\mathbf{x}) : \|\boldsymbol{\beta}\|_2 \leq 1/\rho,\ \|\mathbf{x}\|_2 \leq r\}$:*

$$\mathcal{R}_{\mathcal{D}_T}(f) \quad \leq \quad \frac{1}{N} \sum_{i=1}^{N} \log_{e_0} \left(1 + e^{-2y_i \boldsymbol{\beta} \cdot \mathbf{x}_i}\right) + D_{\mathcal{H}_0,\tilde{\mathcal{H}_0}}(\mathcal{D}_S, \mathcal{D}_T) \tag{8}$$

$$+ \frac{16r}{\rho\sqrt{N}} + \sqrt{\frac{\log \log_2 \frac{4r}{\rho}}{N}} + \sqrt{\frac{\log \frac{2}{\delta}}{2N}} \tag{9}$$

where $e_0 = \log(1 + 1/e)$.

5 Experiments

In this section, we evaluate our method on various real-world datasets. Through experiments, we argue that our method achieves better classification results compared to the existing methods within a lower computation time. In addition, we verify that even when the distributions of published data and encrypted data are very different, our method can properly correct the difference and maintain the classification performance.

Datasets We used five widely-used classification datasets from UCI data repository - the adult income dataset (**Adult**), the bank marketing dataset (**Bank**), the Wisconsin Breast Cancer dataset (**Cancer**), the Pima Indians Diabetes dataset (**Diabetes**)and the Australian Credit Approval (**Credit**) dataset [9]. **Adult** and **Bank** datasets are commonly used in fair machine learning studies, so we treat variables that can induce social bias, such as gender and age, like private variables. For other datasets, the variable that has the greatest impact on the classification performance was selected as a private variable. In practical applications, data owners can flexibly set private variables according to their own security standards. The explanation for each dataset is shown in Appendix.

Experimental Setup All the experiments were done on a machine equipped with 40 threads of an Intel Xeon E-2660 v3 @2.60GHz CPU processor. We implemented step 1 of our framework with Python 3.6.3, using the LR module in the scikit-learn library. Other steps were implemented with C++, using HEAAN v1.1 [4] for HE. HEAAN is an implementation of the CKKS scheme, a detailed

Table 1: Classification Results on five datasets.

Dataset	Accuracy (%)			Computation time (sec)			Time per iteration (sec)	
	LRHE	Ours	Ours-grad	LRHE	Ours	Ours-grad	LRHE	Ours-grad
Adult	81.913	**83.123**	82.759	306.061	**102.25**	1488.65	58.372	**55.786**
Bank	89.712	89.823	**89.934**	248.854	**82.315**	1095.44	47.528	**43.184**
Cancer	86.957	**90.580**	**90.580**	225.286	**73.392**	968.019	42.419	**37.791**
Diabetes	71.429	75.325	**76.623**	184.733	**61.183**	700.522	35.038	**30.927**
Credit	97.794	98.453	**98.529**	201.763	**65.344**	828.833	38.415	**34.757**

description of the scheme and its parameters is in Appendix. We used privacy-preserving LR in [18] (**LRHE**) as a comparison method because it is the only MLED training model which works in a practical computation time. We don't compare our method with the HE-friendly SVM model in [28], because their model assumes that the data owners pre-compute the kernel matrix, which can leak some information about the model. Comparing our model with DNN models in [22] is not in our interest because they have too high latency.

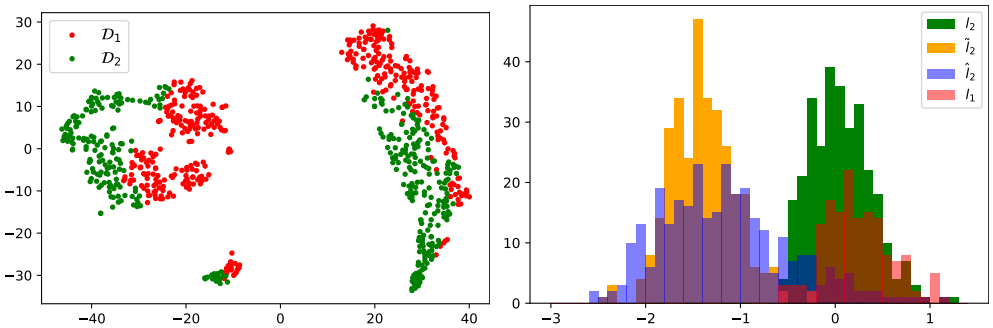
For each dataset, we randomly sampled 20% as test samples, and 20% Of the other 80% was treated as plaintext data and used for the training of step 1. We encrypted the private variable of the remaining 60%, and used it for step 2 and 3. For CKKS parameters, we used $N = 2^{16}, q_L = 2^{1200}$ and $P = 2^{40}$. The approximation degree of sigmoid function for **LRHE** was set to 3, because increasing the degree results in a larger multiplicative depth and the less possible number of gradient descents with LHE. Also, it was observed in our experiments that increasing the degree up to 7 didn't affect the performance of the model significantly. The learning rate for **LRHE** was chosen in $\{0.001, 0.0001, 0.00001\}$ to achieve the best classification performance.

Results We compared the methods in terms of their classification accuracy and computation time. The results of the experiments are summarized in Table 1. For all the datasets, **Ours** achieves higher accuracy and lower latency than **LRHE**. Especially for the computation time, **Ours** can reduce latency by 66$\sim$68% through efficient computation using private variables. Although the performance of LR using plaintext is not inferior to ours, the performance of **LRHE** is degraded due to sigmoid approximation and limited iterations of gradient descent. On the other hand, since **Ours** is free from parameter selection, its performance does not decrease compared to the same operation using plaintext.

As an ablation study, we additionally trained the step 3 of **Ours** using gradient descent, with all variables encrypted (**Ours-grad** in Table 1). Because the multiplicative depth per iteration of **Ours-grad** is lower than that of **LRHE** by not using the sigmoid approximation, we trained **Ours-grad** for 8 iterations. The learning rate was chosen in $\{0.001, 0.0001, 0.00001\}$.The procedure that requires the most computation time in **Ours-grad** is a matrix-matrix multiplication, which causes **Ours-grad** to have a greater latency than **LRHE**. However, the computation time per iteration of **LRHE** is higher than that of **Ours-grad**, and the result of the matrix-matrix multiplication can be reused through the iterations after calculating once. Therefore, it is implied that as the number of iterations increases, **Ours-grad** will be more efficient than **LRHE**. Also, the classification performance of **Ours-grad** is similar or higher than that of **Ours**, which means that **Ours-grad** converges better by allowing more iterations than **LRHE**. Therefore, even in situations where partial encryption of private variables is limited, our method has an advantage over the existing method.

Effectiveness of Mean Matching In this section, we verify that the mean matching in the step 2 of our method is a simple but effective way to mitigate performance degradation due to the difference between distributions of unencrypted data and encrypted data. Here we do not randomly divide the dataset, but divide it by the value of a certain variable. **Diabetes** dataset, for example, was divided according to whether the value of 'Glucose' variable was greater than or less than its median. Figure 3(a) plots the dimension reduction result of **Diabetes** using t-stochastic neighborhood embedding, and confirms that the two subsets (namely X_1 and X_2) have very separate distributions from each other. We sampled 20% of the whole dataset from X_1 and trained the step 1 with the samples. 80% of X_2 was used for step 2 and 3, and the other 20% of X_2 was used as test samples.

Table 2 summarizes the experiment results for three datasets, **Adult**, **Diabetes** and **Credit**. For the other two, there was no variable that can properly divide the dataset. The test accuracy of the LR



(a) Visualization of $\mathcal{D}_1$ and $\mathcal{D}_2$.

(b) Logit distribution of $\boldsymbol{l}_1$ (red), $\boldsymbol{l}_2$ (green), $\tilde{\boldsymbol{l}}_2$ (orange). The blue histogram refers to the distribution of logit of X_2 when trained with X_2.

Figure 3: Effectiveness of Mean Matching

Table 2: Ablation study for mean matching. **Criteria** refers to the criteria for dividing the datasets.

Dataset	**Criteria**	**Accuracy (%)**			
		LR with X_1	**Ours-without mean matching**	**Ours**	**LRHE**
Adult	X_1 : 'Marital' = 0 X_2 : 'Marital' = 1	63.330	63.330	69.471	68.690
Diabetes	X_1 : 'Glucose' > 117 X_2 : 'Glucose' $\leq$ 117	29.487	29.487	80.769	80.769
Credit	X_1 : 'A4' = 0 X_2 : 'A4' = 1	20.833	20.833	79.167	79.167

model trained with X_1 is lower than that of **LRHE** trained with X_2, because the domains of X_1 and X_2 are different and the test set was sampled from X_2. The performance of **Ours** without the mean matching is no different from that of the LR with X_1. However, with mean matching the classification accuracy of **Ours** is raised to the same level as that of **LRHE**. Even when the accuracy of LR with X_1 is 50% or lower than that of **LRHE**, applying mean matching completely recovers the classification performance of **Ours**.

To verify the reason why mean matching of our method works well, in Figure 3(b) we plotted the histogram of logits inferred by LR models. In the figure, the red and blue histograms represent the distributions of logit of X_1 for LR learned with X_1 (=$\boldsymbol{l}_1$) and logit of X_2 for LR learned with X_2, respectively. Thus, the blue histogram can be seen as the distribution of the logit of X_2 when it is classified "correctly". However, because the model is biased to the distribution of X_1 when trained with X_1, the distribution of the logit of X_2 (=$\boldsymbol{l}_2$) is also biased to the distribution of the logit of X_1, as shown in the green histogram. Therefore, mean matching the role of shifting the biased distribution (orange histogram) to fit the correct distribution.

6 Conclusion

We proposed an efficient HE-friendly classification method, which protects both the users' private information and the secrecy of the model. We trained a Ridge regression model for the logit instead of logistic regression, which has a closed-form solution and is free from parameter search. To extract logit from the binary label, we trained a teacher model on unencrypted data that can output logit, similar to that of knowledge distillation. By encrypting only private variables that require a high level of security instead of encrypting all information, our method can achieve higher efficiency and training stability. Our algorithm is HE scheme-free in the sense that it can bring efficiency when implemented with any HE scheme that is widely used.

References

[1] Y. Aono, T. Hayashi, L. T. Phong, and L. Wang. Input and output privacy-preserving linear regression. *IEICE TRANSACTIONS on Information and Systems*, 100(10):2339–2347, 2017.

[2] A. Ben-Israel. An iterative method for computing the generalized inverse of an arbitrary matrix. *Mathematics of Computation*, pages 452–455, 1965.

[3] Y.-R. Chen, A. Rezapour, and W.-G. Tzeng. Privacy-preserving ridge regression on distributed data. *Information Sciences*, 451:34–49, 2018.

[4] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 409–437. Springer, 2017.

[5] J. H. Cheon, K. Han, A. Kim, M. Kim, and Y. Song. Bootstrapping for approximate homomorphic encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 360–384. Springer, 2018.

[6] E. Christodoulou, J. Ma, G. S. Collins, E. W. Steyerberg, J. Y. Verbakel, and B. Van Calster. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of clinical epidemiology*, 110:12–22, 2019.

[7] M. d. Cock, R. Dowsley, A. C. Nascimento, and S. C. Newman. Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2015.

[8] A. De Caigny, K. Coussement, and K. W. De Bock. A new hybrid classification algorithm for customer churn prediction based on logistic regression and decision trees. *European Journal of Operational Research*, 269(2):760–772, 2018.

[9] D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

[10] K. Gai, M. Qiu, and H. Zhao. Privacy-preserving data encryption strategy for big data in mobile cloud computing. *IEEE Transactions on Big Data*, 2017.

[11] R. E. Goldschmidt. *Applications of division by convergence*. PhD thesis, Massachusetts Institute of Technology, 1964.

[12] R. M. Gower and P. Richtárik. Linearly convergent randomized iterative methods for computing the pseudoinverse. *arXiv preprint arXiv:1612.06255*, 2016.

[13] S. Halevi. Homomorphic encryption. tutorials on the foundations of cryptography. isc, 2017.

[14] S. Halevi and V. Shoup. Bootstrapping for helib. In *Annual International conference on the theory and applications of cryptographic techniques*, pages 641–670. Springer, 2015.

[15] M. Jagielski, M. Kearns, J. Mao, A. Oprea, A. Roth, S. Sharifi-Malvajerdi, and J. Ullman. Differentially private fair learning. In *International Conference on Machine Learning*, pages 3000–3008. PMLR, 2019.

[16] N. Kilbertus, A. Gascón, M. Kusner, M. Veale, K. Gummadi, and A. Weller. Blind justice: Fairness with encrypted sensitive attributes. In *International Conference on Machine Learning*, pages 2630–2639. PMLR, 2018.

[17] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC medical genomics*, 11(4):23–31, 2018.

[18] M. Kim, Y. Song, S. Wang, Y. Xia, and X. Jiang. Secure logistic regression based on homomorphic encryption: Design and evaluation. *JMIR medical informatics*, 6(2):e19, 2018.

[19] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.

[20] L. Lombardo and P. M. Mai. Presenting logistic regression-based landslide susceptibility results. *Engineering geology*, 244:14–24, 2018.

[21] Q. Lou and L. Jiang. She: A fast and accurate deep neural network for encrypted data. *arXiv preprint arXiv:1906.00148*, 2019.

[22] Q. Lou, B. Feng, G. C. Fox, and L. Jiang. Glyph: Fast and accurately training deep neural networks on encrypted data. *arXiv preprint arXiv:1911.07101*, 2019.

[23] Q. Lou, W.-j. Lu, C. Hong, and L. Jiang. Falcon: Fast spectral inference on encrypted data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2364–2374. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/18fc72d8b8aba03a4d84f66efabce82e-Paper.pdf`.

[24] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1): 3–es, 2007.

[25] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning*. MIT press, 2018.

[26] T. Morshed, D. Alhadidi, and N. Mohammed. Parallel linear regression on encrypted data. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–5. IEEE, 2018.

[27] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft. Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, pages 334–348. IEEE, 2013.

[28] S. Park, J. Byun, J. Lee, J. H. Cheon, and J. Lee. He-friendly algorithm for privacy-preserving svm training. *IEEE Access*, 8:57414–57425, 2020.

[29] G. Qiu, X. Gui, and Y. Zhao. Privacy-preserving linear regression on distributed data by homomorphic encryption and data masking. *IEEE Access*, 8:107601–107613, 2020.

[30] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[31] M. Sumathi and S. Sangeetha. Enhanced elliptic curve cryptographic technique for protecting sensitive attributes in cloud storage. In *2018 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, pages 1–5. IEEE, 2018.

[32] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[33] Q. Wang and X. Shi. (a, d)-diversity: Privacy protection based on l-diversity. In *2009 WRI World Congress on Software Engineering*, volume 3, pages 367–372. IEEE, 2009.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] We defer the proofs to Appendix
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Include them in the supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] We used only the publicly available datasets
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]